UNIVERSITY OF GOTHENBURG

Exploring an architecture for extending vehicle telematics Services to mobile devices:

A Design Research approach

Bachelor of Science Thesis in Software Engineering and Management

RAN TSOREF
JON FRIDÉHN
KIMMI SANDHU

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, June 2011

Exploring an architecture for extending vehicle telematics services to mobile devices:
A Design Research approach

Ran Tsoref
Jon Fridéhn
Kimmi Sandhu

Examiner: Helena Holmström Olsson

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2011

BACHELOR THESIS

Exploring an architecture for extending vehicle telematics services to mobile devices: A Design Research Approach

Ran Tsoref
tsoref@ituniv.se

Jon Fridéhn
jon.fridehn@gmail.com

Kimmi Sandhu
kimmi.madeline@gmail.com

IT University of Gothenburg, Software Engineering and Management. Gothenburg, Sweden

Abstract

In the vehicle industry there has been a constant transformation from mechanical and hydraulic control systems to computer-based systems over the last two decades; this transformation contributed to the foundation of telematics systems. The main objective of this report is to explore how the current generation of mobile devices may interact with embedded and purpose-built systems. We develop a prototype application that uses wireless two-way communication for remote interaction, and based on a literature study where appropriate quality attributes are identified, we reflect on the design experience and suggest an architecture for similar telematics-to-mobile devices solutions.

Keywords: android, mobile devices, telematics, vehicle on-board system.

1 Introduction

Over the last two decades the vehicle industry has been constantly transforming the mechanical and hydraulic control systems to computer-based systems. This transformation together with the dependency of a large population on content, contributed to the foundation of telematics (Juliussen 2003). As the number of electronic systems increases, computer controlled vehicle data can provide added value to the owners by giving them more detailed status information of their vehicle (Juliussen 2003; Karimi et al. 2004). The information gathered from the vehicle's integrated electronic systems and sensors is presented on the vehicle's dedicated display panels, or has to be accessed using specialized equipment and software connected directly to the vehicle. In both cases the user only receives limited information of the actual status of the car. As discussed by Kuschel (2008), vehicle services are only accessible through the vehicle itself, despite the fact that there is a demand for accessing the vehicle sensor data remotely. Furthmore, Kuschel (2008) describes remote vehicle diagnostics as a topic that the vehicle industry aims at achieving, as this is yet not achieved.

Current generation of mobile devices such as smartphones and tablet PCs are a rapidly growing trend (Mahmoud, 2008; Seada, 2010). These mobile devices are designed to communicate in different communication technologies (Wu et al. 2009) and are equipped with relatively powerful processors (Vaughan-Nichols 2003). Such mobile devices may act as complements to the dedicated display, thus giving the user more elaborated information as well as allow remote access.

Responding to this, we explore how the current generation of mobile devices may interact with embedded and purpose-built systems. Specifically, we take two steps to address this research objective. [1] We develop a prototype application (KITT[1]) that uses wireless two-way communication for remote interaction. This prototype is developed in collaboration with two industry partners, one consultant company specializing in diagnostics, telematics and mobile systems solutions and one subsidiary to a large boat company providing safety-critical electronic control systems. [2] Based on a literature study where appropriate quality attributes are identified, we reflect on the design experience and suggest an architecture (MDVI - Mobile Device Vehicle Interaction) for similar telematics-to-mobile devices solutions. To achieve this we take a design research approach where the prototype acts the main practice contribution. The suggested architecture is intended both as a practice contribution and research contribution given the influence of related research and experience from the prototype development.

This report is organized as follows: section 2 presents a problem domain overview, the research approach used, and explains the different steps carried out for the research. Section 3, 4, and 5 corresponds to development iterations 1, 2, and 3 respectively. All of these sections show the continuous progress of both the architecture design and the implementation of the prototype. In addition to this, section 3 also presents data gathered from the literature study and section 6 includes an evaluation of the architecture. The last section concludes our findings and discusses further research.

2 Research approach

This section gives an overview of the problem domain as well as describes the research approach that was used in this research study and how it was adapted to fit this research. Below we present how we use design research to investigate this problem domain. Section 2.2 describes design research as it is defined by Vaishnavi & Kuechler (2005) while section 2.3 describes our actual process followed.

2.1 Problem domain overview

This study combines the telematics domain with mobile technology to understand how they may interact. To gain an overview of the problem domain we consulted previous and related research done in the same domain.

Karimi et al. (2004) examines how software architecture for an aftermarket remote vehicle diagnostic (RVD) system should be designed. They present an architecture based on several different design principles and a proof-of-concept prototype implemented using the designed architecture.

Gehlen et al. (2006) describes a vehicle-to-infrastructure (e.g. Internet) architecture. They present a vehicle communication gateway (VCG), positioned in the car, to which a user can connect using different mobile devices (e.g PDA) with e.g. Bluetooth or WLAN. The VCG implements the IEEE 802.21 specification and the Media Independent Handover (MIH) to be able to choose the best available network according to the user's preferences, operator con-

[1] K.I.T.T. is the computerized talking car from the series of Knight Rider (1982). In the original series, K.I.T.T. stands for "Knight Industries Two Thousand".

tracts, etc., and connects to the internet/portal using a long range communication system such as UMTS, GPRS, and WLAN.

Spelta et al. (2010) describes an interaction system (VEDE) for motorcycles, using Bluetooth technology, with vehicle-to-driver communication. The driver use voice instructions through an integrated audio system in the helmet and a vehicle-to-environment communication for remote maintenance and diagnostics. The vehicle can also send audio messages back to the driver. The authors have designed a four layer software architecture, representing the vehicle-to-driver interaction. The third layer is represented by a smartphone, which communicates with the helmet (forth layer) and a car CAN (Controller–area network) Bluetooth gateway/converter (second layer). The motorcycle (first layer) is where the vehicle control unit (VCU) collects data and manages the vehicle. The VEDE system was implemented and evaluated using a hyper-sport motorcycle, a "Bluetooth"-helmet and two different smartphones.

All of the related research has parts that highlight the objective of our work. Karimi et al. (2004) and Spelta et al. (2010) both present architectures that give a good basis for designing our architecture as well as giving hands on experience regarding communication with vehicles. Spelta et al. (2010) also provide a specific solution on how a mobile device can be used as a communication gateway between a driver and a vehicle. These three provide a useful introduction to vehicle-to-infrastructure communication, which will be later extended in our iterative research approach.

2.2 Design research

As presented by Vaishnavi & Kuechler (2005), analyzing the use and performance of designed artifacts, help in understanding, explaining or improving the behavior of aspects in information systems. These are all important concepts of design research approach. There are five different phases in design research described by Vaishnavi & Kuechler (2005), which are illustrated in figure 2.1 and briefly explained below.

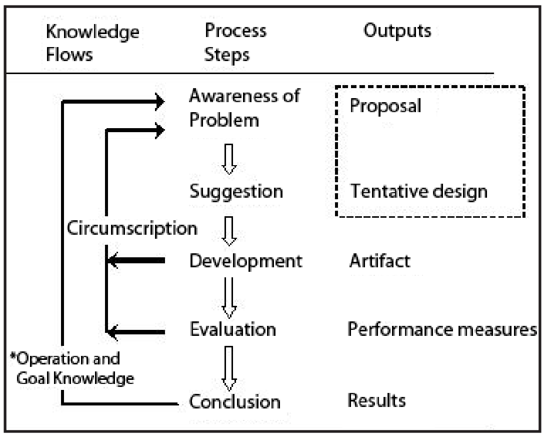


Figure 2.1 *Design Research General Methodology (Vaishnavi & Kuechler 2005)*

Awareness of problem: There are different ways of getting to grips with a problem; one suggestion may be to read in a related practice to understand the research field. This phase returns a proposal as an output to a new research effort.

Suggestion: This is the phase where new functionality is anticipated after having the proposal output from the previous phase, where existing or new elements are outlined. It's important not to settle with a proposal if it doesn't make sense after deliberation, but instead put it aside, and considered it at a later time. If no other proposal comes up during the Suggestion phase one can revisit the Awareness of Problem phase to try to find another problem to consider.

Development: The artifact that need to be constructed is implemented in this phase, where the result will vary depending on the design of the artifact. It's important to understand that this phase is not always straightforward, since development can bring to light new issues or improvements that affect the overall output. If something doesn't make sense during implementation there may be a need to go back to the Suggestion phase and iterate before moving on.

Evaluation: After developing the artifact (i.e the architecture) it is evaluated according to what criteria was presented in the proposal from the Awareness of Problem phase. The behavior of the artifact is hypothesized about in this phase and is cautiously explained while either confirming or contradicting the hypothesis. The results of this phase are together with any additional information gone through another iteration of the Suggestion phase.

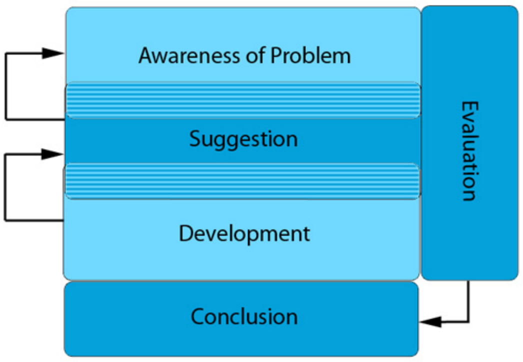


Figure 2.2 *Adapted Design Research Strategy*

Conclusion: The results of the research effort are written up in this phase which mostly is spent with categorizing the knowledge between “firm” facts and “loose ends”, the latter could be used for further research.

2.3 Research design

The five phases of the design research strategy suggested by (Vaishnavi & Kuechler 2005) are further adapted to suit this study as illustrated in figure 2.2.

Awareness of problem: Our industry partners emphasize that the introduction of vehicle telematics and its ability to communicate using wireless technologies brings new opportunities for owners of current generation mobile devices, such as smartphones or tablet PCs. This would allow the user to remotely connect to their vehicle and both receive and send information.

This phase was carried out as a literature study where the research field was being explored to gain more understanding and knowledge of the problem. The search criterion was based on keywords from the mobile platforms, telematics, software architecture, and wireless communication areas. These areas were found to be the most relevant for carrying out our research objective. Similar to how Runeson & Höst (2009) argue that data analyzed in parallel with data collection creates opportunity to take new insights into consideration, we have approached our literature study in increments and continuously analyzed data while searching for new sources.

Suggestion: Based on the literature study conducted in the previous phase, a preliminary architecture was designed. As the research approach is iterative in its nature, output and new insights from Awareness of Problem and Development phases were

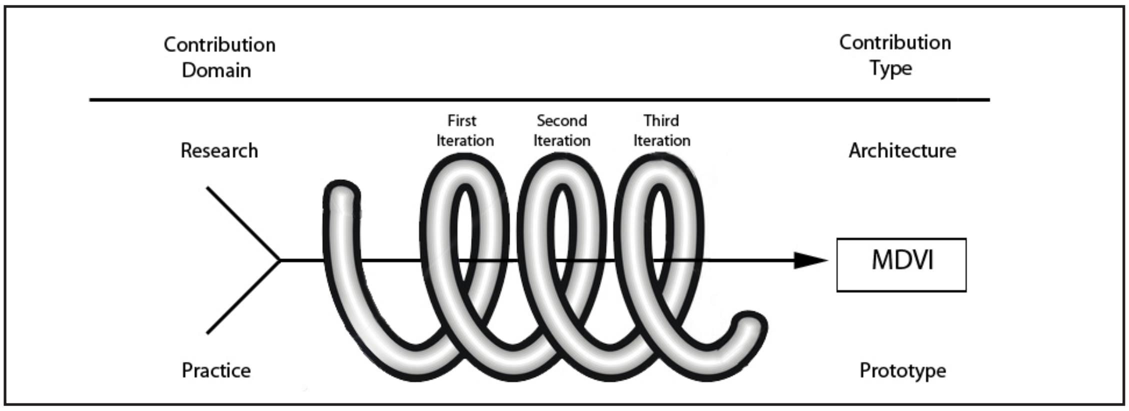


Figure 2.3 *Iterative research process*

used as inputs to this phase in order to design and further improve the architecture.

Development: In this phase we used the architecture designed in the Suggestion phase as a basis for implementing an Android prototype application, which illustrates the design in practice. Each iteration further improves and implements new changes to the prototype and subsequently also to the architecture. As can be seen in figure 2.4, which is an illustration of our process, the three iterations can clearly be distinguished. It is also visible in our process how working through research and practice in this sort of iterative manner, subsequently gives an architecture and a prototype as our contribution at the end of the third iteration.

Evaluation: As Vaishnavi & Kuechler (2005) suggests, the evaluation process is being continuously performed by the researchers throughout the different phases in form of mini evaluations. However, they are not negating the need for conducting the evaluation phase and therefore, while implementing the prototype we iterated and evaluated the parts that correspond to the architecture in each iteration. Following Vaishnavi & Kuechler (2005) suggestion, we noted any deviations from our expectations, and the output from each evaluation phase was used to make hypothesis about the architecture and helped in redesigning it. After the prototype development was finalized we evaluated the architecture based on the implemented application and the literature findings.

The MDVI architecture is evaluated and created by the influence of both the practise and research sides of the coil, which is visible in figure 2.3.

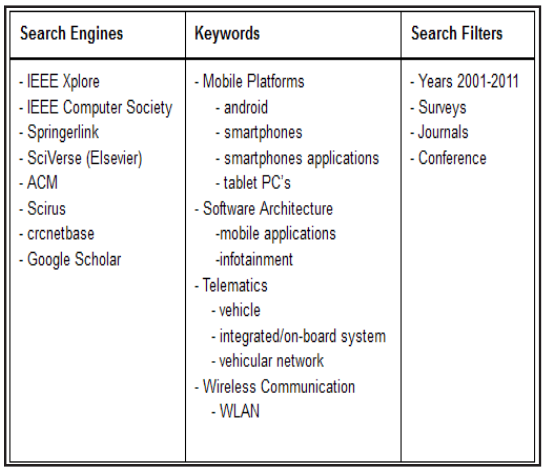
Conclusion: After the evaluation phases in the three iterations, the architecture was contrasted with the study objective, from which we drew conclusions regarding whether our research can be categorized as what Vaishnavi & Kuechler (2005) specifies as either "firm" or "loose ends". From the final conclusion we also determined if further research could be carried out and what would be possible paths for it to take, considering the different findings in the study.

3 First iteration

In this section we present the first iteration conducted in this study. We discuss findings from the literature study that are related to our research in section 3.1 and present the first draft of the MDVI architecture in section 3.2. In section 3.3 we discuss the development of the prototype in its initial stage. As the prototype application is implemented for a Google's Android device and is communicating via WLAN, we put more focus on the explanation of these concepts.

3.1 Awareness of problem: Literature study

The MDVI architecture design is based on both the development of the KITT and a literature study, for which the search criteria can be visualized in table 3.1.



Search Engines	Keywords	Search Filters
- IEEE Xplore - IEEE Computer Society - Springerlink - SciVerse (Elsevier) - ACM - Scirus - crcnetbase - Google Scholar	- Mobile Platforms - android - smartphones - smartphones applications - tablet PC's - Software Architecture -mobile applications -infotainment - Telematics - vehicle - integrated/on-board system - vehicular network - Wireless Communication - WLAN	- Years 2001-2011 - Surveys - Journals - Conference

Table 3.1 *Search criteria for the literature study*

3.1.1 Vehicle telematics systems

Telematics is referred to as the convergence of telecommunication and informatics. It provides a vehicle with abilities such as wirelessly both send information regarding the state of the vehicle and receive and process information from its surroundings, in addition to GPS capabilities.

When it comes to diagnosing a vehicle, telematics can make available Electronic Control Unit (ECU) information to remote diagnostic servers which in turn can make a better assessment if there is some vehicle complication.

By having a two-way communication, the telematics system can constantly receive new information regarding e.g. traffic conditions and adjust its recommended route to take, in order to make the drivers journey safer and more pleasant (Grymek et al. 2007). Further safety precautions e.g. distress signals are sent automatically or manually to an emergency center or a call center, in the case of a more severe vehicle breakdown (Karimi et al. 2004; Grymek et al. 2007).

Two other areas of telematics are infotainment and entertainment which e.g. can provide Internet access or information regarding the nearest point of interest (POI), e.g. gas station or restaurant (Karimi et al. 2004).

3.1.2 Vehicle integrated wireless communication technologies

Recent research (Nolte et al. 2005; Vassilaras & Yovanof 2010; Hossain et al. 2010), present wireless communication technologies and protocols that are integrated or being adopted in vehicles as part of the effort for developing what the literature refer to as Intelligent Transportation Systems (ITS) (Vassilaras & Yovanof 2010; Hossain et al. 2010; Juliussen 2003). Two main concepts that are derived from the ITS are: In-vehicle communication (communication within the vehicle with portable devices e.g. mobile phones and laptop computers) and Inter-vehicle communication (communication between the vehicle and the surrounding environment) (Nolte et al. 2005). These two concepts are shown by the literature as the main driving force behind the integration of wireless communication technologies as part of vehicle systems (Vassilaras & Yovanof 2010; Hossain et al. 2010).

Wireless technologies that are recognized by researchers as integrated or future to be adopted in vehicles includes Bluetooth, ZigBee, UWB (IEEE 802.15.3a), Wi-Fi (IEEE 802.11), WAVE (IEEE 802.11p) and cellular approaches e.g. 3g and GPRS (Nolte et al. 2005; Vassilaras & Yovanof 2010; Hossain et al. 2010; Hsu et al. 2005).

3.1.3 Android platform and its architecture

At present there are three major mobile device platforms used in smartphones and tablet PC's (Gartner 2011)[2]. In this paper we are focusing on creating the architecture generically for mobile device platforms, e.g. Android, Windows Phone 7 and iOS.
Since the prototype is implemented in Android, the Android platform's architecture is described thoroughly.

The Android operating system's architecture consists of five layers that are divided in different components (Android Developers 2011). First and foremost, it starts at the top with the Applications layer, which can be visible in figure 3.1. The Applications layer consists of the core applications that Android is shipped with. All applications are written in the Java programming language.

The next layer in the Android architecture is the Application Framework layer. Since the Android platform is an open development platform, the developers are able to access and take advantage of the framework APIs[3] that is used by the core applications. The components are encouraged to be reused, and that is visible in the architecture design. The Libraries layer in the architecture in-

[2] Statistics taken from Gartner, (www.gartner.com)

[3] Application Programming Interface consists of sets of standardized requests that allow different computer programs to communicate with each other. APIs establish the proper way for a developer to request services from a program (Britannica Online Encyclopedia 2011).

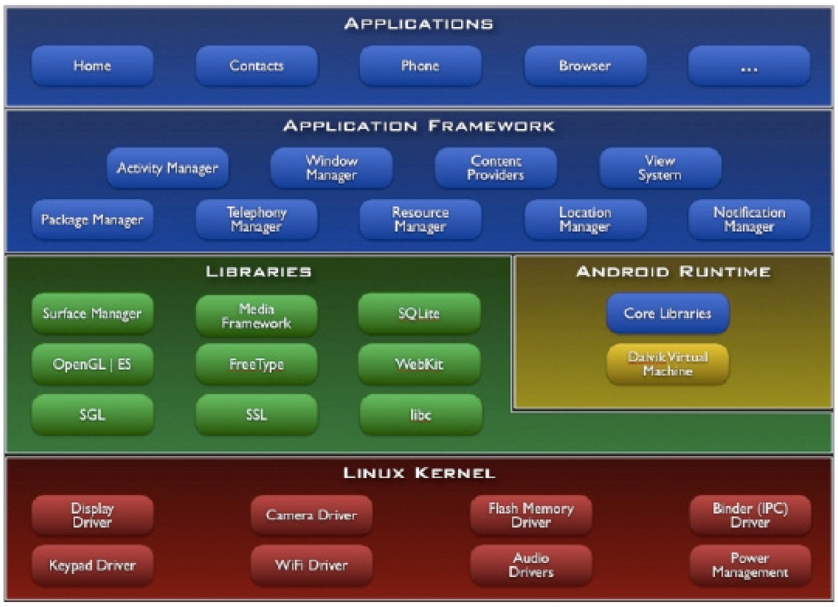


Figure 3.1 *Android Architecture*

cludes a set of C/C++ libraries and these are visible to the developers via the Android application framework. There are also a set of core libraries included in Android that provides functionality that is derived from the Java language's core libraries.

During run-time, every Android application runs in an own process, with an own instance of the Dalvik Virtual Machine. The Dalvik VM uses the Linux kernel for bringing the underlying functionality like low-level memory management and threading. The last layer in the Android architecture is the Linux Kernel layer, which is responsible for the core system services such as security, memory management, process management, network stack, and driver model. The Linux Kernel is also working as an abstraction layer between the hardware and the software (Android Developers 2011). The complete Android architecture is displayed in figure 3.1.

When relating the Android platform with the OSI Model[4], we find that Google's Android provide in their API libraries, the ability to develop an application where the implementation starts at the third layer of the OSI Model i.e. the Network layer, with abilities for utilization of socket programming (Android Developers 2011).

[4] The OSI model (Open System Interconnection model) is sub-dividing a communications system into seven different layers. Each layer provides services to the layer above and receives services at the same time from the layer below (Day & Zimmermann 1983).

3.1.4 Mobile devices communication technologies

Some of the communication technologies used in modern mobile devices includes Wi-Fi, WiMax, GPRS and Bluetooth (Wu et al. 2009). These communication technologies can connect the mobile device wirelessly with other communication devices, e.g. a GPS system in a vehicle. According to Canalys Q2 2008 Statistics (Canalys 2008), the mobile devices that are distributed in Europe, Middle East and in Africa, have a 38% rate on having a GPS technique integrated in the smartphone (Pei et al. 2009).

Integrated communication technologies in smartphone devices is a key feature that is needed to further develop the in-vehicle communication and the inter-vehicle communication for the future of wireless interaction. At the present time, smartphone devices use some form of cellular wireless standard, such as 3G or 4G to provide the smartphones with Internet access (ITU 2011).

3.1.5 Wireless Fidelity (IEEE 802.11)

As requested by our industry contact, the application prototype was implemented to communicate via WLAN. As it was evident from the literature, Wi-Fi (Wireless Fidelity) is a common wireless technology standard integrated in modern mobile devices (Wu et al. 2009). Wi-Fi refers to any of the IEEE 802.11 network standards, e.g. 802.11a, 802.11b, 802.11g (Nolte et al. 2005; Dhawan 2007).

The range of Wi-Fi is around 50-100 meters indoor and up to 1000 meters outdoor, this will however depend a lot on the surrounding conditions (interference, etc.) and performance will also decrease with distance (Dhawan 2007). According to the IEEE 802.11n specifications, transfer speed of up to 600 MB/s is supported and it runs on the 2.4 Ghz or 5 Ghz spectrum. Wi-Fi works with so called Access Points (AP) which wirelessly sends out beacons (packets with the Service Set Identifier (SSID) which is the network name), a client use this packets information to connect to the AP (Dhawan 2007).

3.2 Suggestion: Architecture first draft

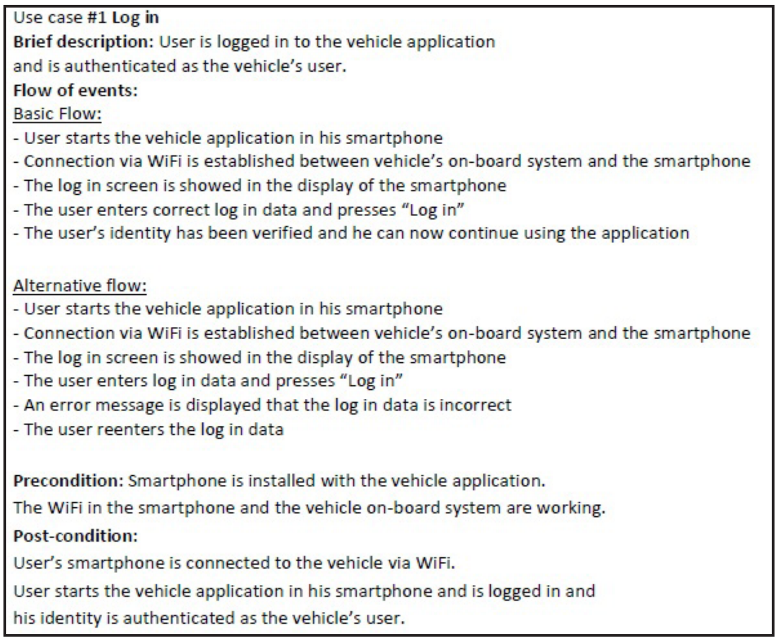
3.2.1 Defining the stakeholders

While there may exist additional stakeholders, this study focuses on two: user and service technician (figure 3.3). The user of a vehicle is the main User of the application, and is likely the most interesting from a business perspective. The Service Technician is a secondary administrative user who can access more technical information of a vehicle to be able to save this information to his mobile device without the need of specialized equipment. For example, the service technician is able to extract the error message log to quickly get an overview of the vehicle status and determine if any special services are required. The focus on these two further matches the main interest of our industry collaborators.

3.2.2 Identifying the quality attributes

The first step in making the generic architecture for a mobile device application that is connected to a vehicle integrated system is to identify the strengths that the architecture would have, i.e. the quality attributes. When identifying the major quality attributes of the architecture, the most common way to start with it is to make scenarios which describe the system's behavior in more detail (Bass et al. 2003). Hence, the scenarios help to describe the qualities of the system, and the architectural tactics describe how a certain quality can be achieved (Bass et al. 2003). To identify which quality attributes were relevant to the MDVI architecture we wrote a number of use case scenarios for the application; one scenario is illustrated in figure 3.2. The complete set of scenarios can be found in Appendix A of this study. Through the use case scenarios we identified the following quality attributes: security and usability.

Figure 3.2 shows the use case for logging in to the application as a user of the vehicle. By writing this use case scenario, we identified the need for security as a quality attribute. Since the application is communicating with an external source and holds credential about the users, we considered security as an important quality attribute for the MDVI architecture.
With the thought of having two user types (user and service technician) for the application, we considered the users' need to be exposed to distinct menus and information. The user should be able to navigate easily to the main functionality of an application, which is to get information from the vehicle (please refer to Appendix A, use case scenario no. 2). Hence, usability is an important quality attribute to the MDVI architecture, as it aims in easing the user experience with an application. As the MDVI architecture is generic platform wise, its com-


Use case #1 **Log in**
Brief description: User is logged in to the vehicle application and is authenticated as the vehicle's user.
Flow of events:
Basic Flow:
- User starts the vehicle application in his smartphone
- Connection via WiFi is established between vehicle's on-board system and the smartphone
- The log in screen is showed in the display of the smartphone
- The user enters correct log in data and presses "Log in"
- The user's identity has been verified and he can now continue using the application

Alternative flow:
- User starts the vehicle application in his smartphone
- Connection via WiFi is established between vehicle's on-board system and the smartphone
- The log in screen is showed in the display of the smartphone
- The user enters log in data and presses "Log in"
- An error message is displayed that the log in data is incorrect
- The user reenters the log in data

Precondition: Smartphone is installed with the vehicle application.
The WiFi in the smartphone and the vehicle on-board system are working.
Post-condition:
User's smartphone is connected to the vehicle via WiFi.
User starts the vehicle application in his smartphone and is logged in and his identity is authenticated as the vehicle's user.

Figure 3.2 *Use case scenario for logging in*

ponents should be reusable in constructing similar applications for communication with embedded and purpose-built systems. Thus, reusability is a quality attribute that we did not identify from the use scenarios, but considered as important for our architecture design and thus include it from this point and onwards. This means our architecture is guided by three quality attributes: security, usability, and reusability.

3.2.3 Determining the architectural pattern

The three-tier pattern is a client-server architecture which consists of three independent layers, which are the user interface, business logic and database. Since the three tiers are independent, each of them can be replaced or upgraded without affecting the other tiers. The information from the vehicle integrated systems needs to be stored in the application. The most efficient way to store data is within a database. A common architecture model for such a client-server structure is the three-tier software architecture pattern. A benefit of the three-tier architecture is that it is designed to handle security issues since the tiers are separated.

The three-tier architecture together with the stakeholders is illustrated in figure 3.3.

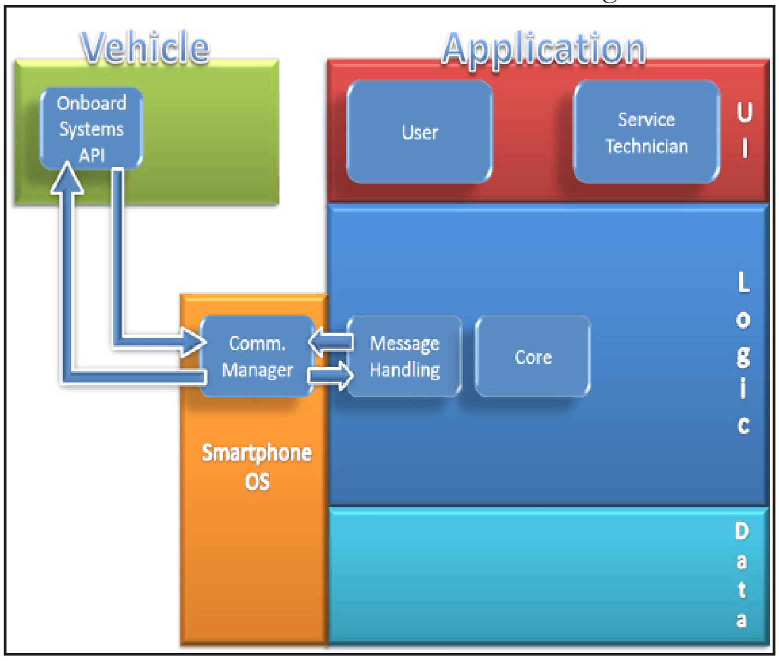


Figure 3.3 *Architecture First Iteration*

3.2.4 Communicating with the vehicle

All modern vehicles have a communication protocol which can be encoded and decoded to pass messages to and from a vehicle (Hubaux et al. 2004). In order to communicate with a vehicle, we added a separate Message Handling module for encoding and decoding vehicle messages. Subsequently, we realized that we could utilize the Android OS's Communication Manager to communicate with the vehicle's protocol via Wi-Fi with the IEEE 802.11g standard. Thus, there are three modules involved in the communication between a vehicle and an application. The vehicle itself has a communication protocol that can be linked with the mobile device's OS communication manager which in turn needs to be encoded and decoded in the application itself. This interaction is illustrated in figure 3.3.

3.3 Development: Prototype

Following the walking skeleton approach (Cockburn, 2004), we developed the initial version of KITT in order to link together the main architectural components. Through the development, it was evident that the application needed a Core module to handle the start point for running the application. The core module is an intermediary that enables communication between the modules of the application and ensures decoupling. Additionally, it includes menu handling for the different users and will serve as the connection between future modules.

In the first iteration the user interface module was connected to the core module. The message handling module was implemented to receive XML encoded messages from the vehicle, in addition to decoding them into a data format that the application can work with. To enable connection to a vehicle, sockets were set up and basic button navigation was created for the regular user's user interface. Due to the inability of our industry partner to provide us with a simulator, we implemented our own simulator. This simulator was not a part of the prototype, but was needed to simulate vehicle messages and enabled testing of the application's message handling.

4 Second iteration

In this section we present the second iteration conducted in this study. In section 4.1 we discuss changes and additions to the MDVI architecture. New implementation and updates to the prototype are discussed in section 4.2.

4.1 Suggestion: Architecture

4.1.1 Ensuring security

Previous research describes security as an important issue to be considered when interaction with the vehicle system is possible from outside the vehicle (Nolte et al. 2005). To address this issue, we added the Application Security module to the MDVI architecture. This module is responsible for the authentication and ensures that unauthorized access to the application is prevented. Application security module focuses on the security quality attribute, as it was identified as one of the major attributes for the MDVI.

The Application Security module incorporates the authentication of both users and service technicians. When authenticating, the service technicians gain access to more administrative features e.g. restricted log information about the vehicle. In addition to user authentication this module is responsible for authenticating the mobile device with the vehicle gateway and encrypting data transferred from the vehicle. Encryption of this data is important as it prevents misuse of the application and the vehicle it might have access to.

4.1.2 Including vehicle information

In order to include all relevant information of a vehicle we added the Vehicle Monitoring module. As mentioned in literature the existence of ECU's, sensors, and actuators in modern vehicles enable to gather information from these and communicate this information directly to mobile devices (Nolte et al. 2005; Vassilaras et al. 2010; Hossain et al. 2009; Bilchev et al. 2004). The module presents the user with an overview of the vehicle components and information from its sensors that might be relevant for the user. This includes presenting error/notification messages in a readable format on information such as the fuel level, oil level, engine temperature etc.

As we extended the MDVI architecture with the vehicle monitoring module we discovered that two additional modules were of importance for displaying vehicle information, Vehicle Service Diagnostics and Log Manager. The Vehicle Service Diagnostics module is responsible for displaying information relevant to the technician, and includes more technical depth in the information displayed in comparison to the Vehicle Monitoring module. This module is only accessible to the service technician.

Meanwhile, the Log Manager receives all log notification/error messages from the vehicle integrated system and is able to sort them to either a user or a technician. Based on the error codes that are embedded in the messages the Log Manager can redirect the message to either the Vehicle Monitoring (user) or the Vehicle Service Diagnostics (service technician) module. The three modules related to vehicle information are illustrated in figure 4.1.

4.1.3 Building for entertainment

Multimedia and infotainment systems have become an integral part of today's modern vehicles (Vassilaras et al. 2010; Nolte et al. 2005). These systems include features such as media players, multiplayer games

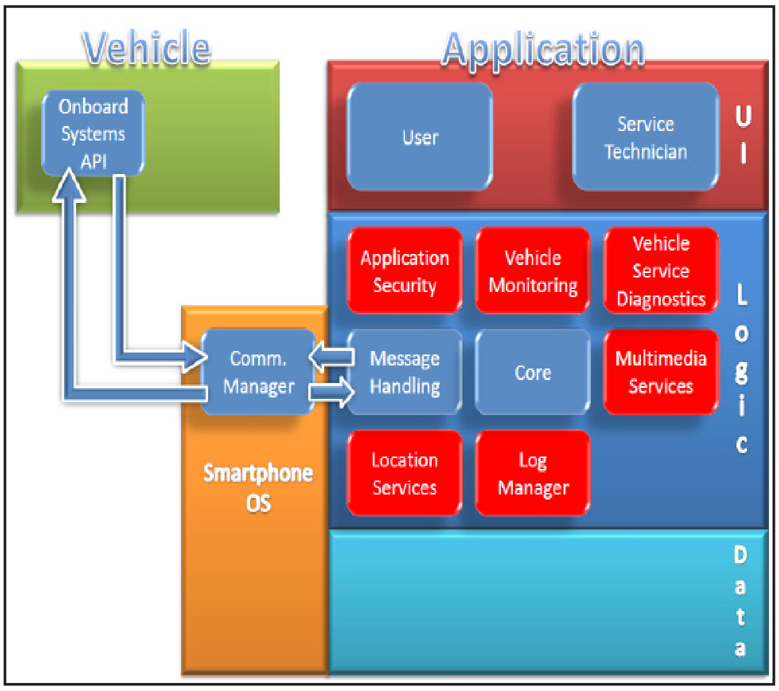


Figure 4.1 *Architecture Iteration 2*

and access to the Internet (Vassilaras et al. 2010; Nolte et al. 2005; Hossain et al 2009; Bilchev et al. 2004). Thus, it was clear that we needed to add Multimedia Services as a separate module in the MDVI architecture. In addition to the Multimedia Services we decided to add Location Services to the architecture, as the use of GPS technology in vehicles is increasing (Vassilaras et al. 2010). This would enable the user to plot and plan routes, then upload or download them to/from the vehicle.

4.2 Development: Prototype

To deal with the security issue we implemented part of the Application Security module in KITT, which we also connected to the Core module. The user interface was updated to present the user with a login window (left image in Appendix C of this study) when the application starts, where the user needs to supply a correct username and password. Depending on the credential the user provides the application distinguish between regular user and service technician and presents menu options accordingly.
Regarding the regular user, in the second iteration we implemented the Vehicle Monitoring module. We needed to present the information from the vehicle in a way that made it easy to survey. This was solved by creating a list for the user interface, showing the different vehicle data (right image in Appendix C of this study). By using a background thread, we were able to use the Message Handling module to continuously request the current values from the vehicle without disrupting or disabling the rest of the application. We also implemented basic functionality, to handle presenting future error/notification messages.

For the service technician functionality we only implemented the base code structure of the Service Diagnostic and connected it to the Core module, as this functionality was not part of the prototype objective but instead intended for future development. The Log Manager was implemented similarly, it was given a base structure and got connected to the Message Handling module, Vehicle Monitoring, and Service Diagnostic. Since both the Vehicle Monitoring and the Service Diagnostic receive messages, they are directly connected to the Log Manager and indirectly connected to the message handling (please refer to Appendix B in this study).

5 Third iteration

In this section we present the final iteration conducted in this study. Section 5.1 discusses the final changes and additions to the MDVI architecture. Last additions and updates made to the prototype are presented in section 5.2.

5.1 Suggestion: Architecture

5.1.1 Communicating with the world

The industrial partners desired the application to be able to send vehicle information to social networks and Google Maps as well as service workshops/centers. This would enable quick diagnostics for the service technicians and may improve service times as they can connect remotely to a vehicle. Additionally, the Multimedia Services needs access to the Internet for some of the purposes of its features. As a response to this, we added the External Services Manager module that was required to handle external communication.

5.1.2 Storing the data

To extend the usage of KITT we decided that previous routes, values and notifications should be accessible with new data in order to visualize statistics over time. It also enables the application to store additional information such as login credentials, pictures or music. Thus, we added the Database Coordinator module to the third tier of the MDVI architecture. The Database Coordinator uses the smartphone operating system's Persistent Storage Manager. The relationship between these two modules is illustrated in figure 5.1.

5.2 Development: Prototype

To understand how the data collected from the vehicle's systems can be used in external services, we implemented the External Service Manager. We used the Google Maps API to implement capabilities to use the gathered data in conjunction with Google maps (please refer to appendix D for a concept of this) and the Facebook API for uploading data to Facebook. The External Service Manager was also connected to the core module and the Database Coordinator.

To enable the data accumulated from the vehicle to be stored in the application, we utilized Android OS's Persistent Storage Manager and implemented a Database Coordinator. We added ability to add, update, fetch and delete data. Since a lot of the other modules benefits from some sort of data storage, the majority of these were connected to the Database Coordinator module (refer to appendix B, MDVI Package Diagram for visualization of this interaction).

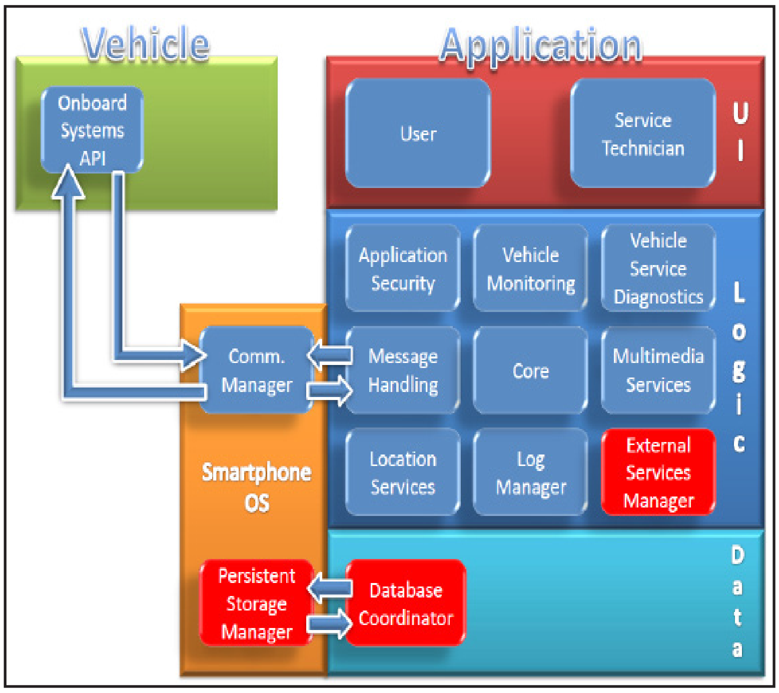


Figure 5.1 *Architecture Iteration 3*

6 Evaluation: The MDVI architecture

Throughout this section we present the contribution of this research. The proposed architecture, Mobile Devices Vehicle Interaction architecture (MDVI). We discuss how an application designed based on the MDVI architecture could be implemented, and which functionalities it could incorporate.

6.1 The composed architecture

Illustrated in figure 5.8 is the entire MDVI architecture, composed in three iterations based on literature study and prototype development (for supplementary diagrams please refer to Appendix B). The MDVI architecture is our main contribution of this research study. While the MDVI architecture focuses only on the use of Android OS, it can be abstracted to the usage of other Mobile Device OS's for the purpose of a generic architecture.

The core functionality of an application based on the MDVI architecture is to help the vehicle user to easy monitor the vehi-

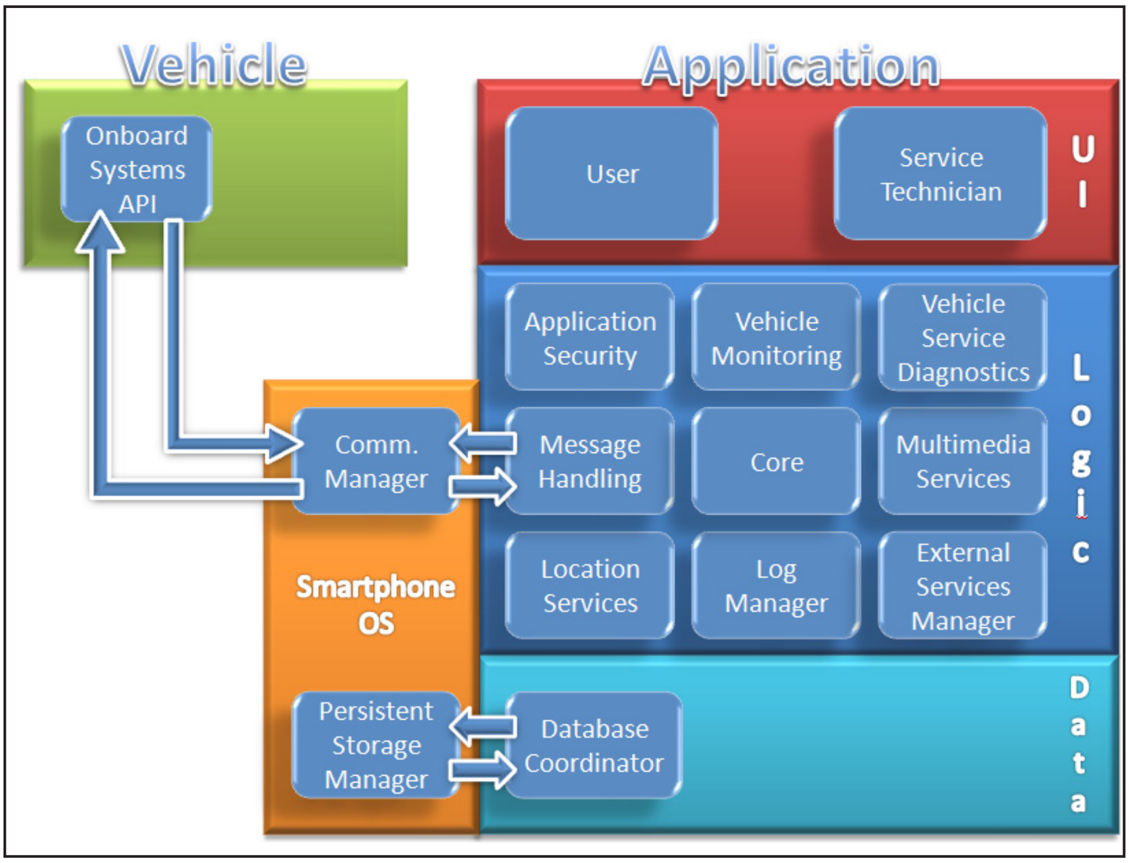


Figure 6.1 *The MDVI architecture*

cle's different inner values e.g. fuel level, boost pressure and water temperature if it is a boat. The user can also receive notifications regarding complications with the vehicle that the user can fix himself, or complications that need to be taken care of by a service technician. The application is able to save the values and notifications, giving the user easy access to this information when he is not near the vehicle. Since the application is meant to be more of an entertainment application, most of the none-core functionality has therefore been geared towards recreation rather than being focused on diagnostics.

By combining the values stored from traveling with input from the phones built-in GPS, the application can plot the travelled route using Google Maps and show how the values changed at different locations (an illustration of this concept can be seen in Appendix D of this study) and also get the average between different locations. The functionality of Google Maps integration is further extended by the ability to plan a route when away from the vehicle and then simply transfer this information to the vehicle once you are connected again. Through integration with social networks such as Facebook, the user is able to upload travelled routes and other information regarding the vehicle to their Facebook page. Furthermore, one could incorporate functionality to find the closest service station, send notifications and log data so that the service station can estimate approximate service cost and book a time for service.

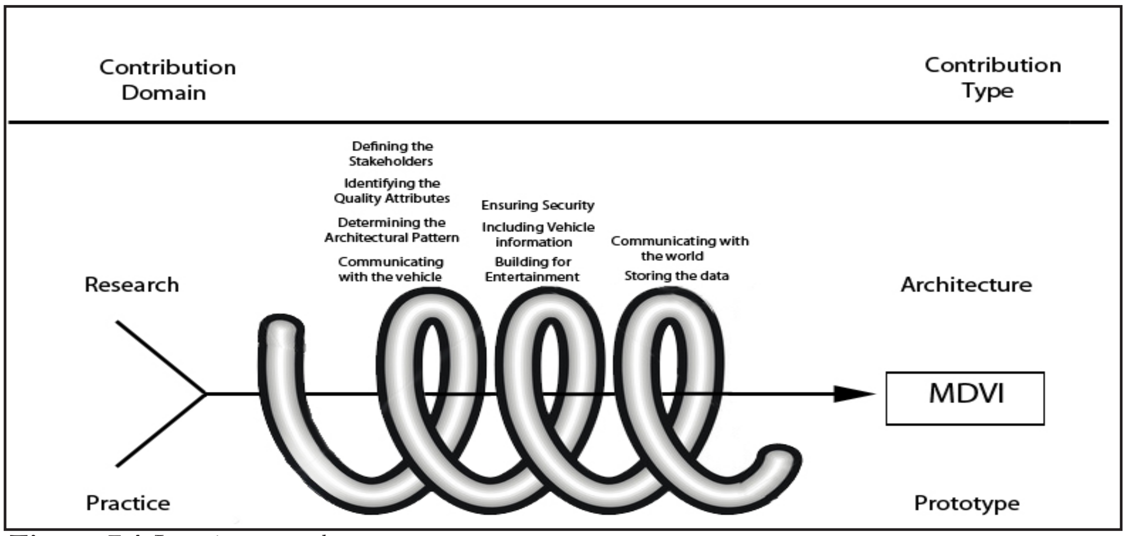
Future extensions to an MDVI based application can include the ability to connect directly to other mobile devices (via peer-to-peer) in order to compare values and other information in real-time between vehicles. As for now, the MDVI architecture is only handling communication between vehicle and mobile device as a direct connection.

Future connections to a vehicle should also be possible indirectly through a server; this scenario is vehicle technology depended, as it requires the vehicle to be connected to a power source and Ethernet cable when parking at the garage or parking lot. Such vision is not far from reality as electric vehicles become more common (Putrus et al. 2009), and so are their charging points (Poul et al. 2009) which can be used for Internet connection if equipped with BPL (broadband over powerlines) technologies (Willie 2006). This vision creates more interesting opportunities including uploading planned routes and check status of vehicle from home, know the location of the vehicle with connection to the vehicle's GPS, and battery level indication for electric cars.

A service mode that will be available to service technicians, extends the entertainment aspect of an application to a more technical one. By logging in and authenticating himself as a service technician, he will be able to, in addition to use regular user functionality, view the full log data, add notes regarding errors and transfer this information to a workshop computer as well as directly order unavailable spare parts that are needed to fix the vehicle.

7 Concluding reflections

The objective of this study was to explore how current generation of mobile devices may interact with embedded and purpose-built systems. We used an adapted Design Research as exploratory approach, drawing on both research and practice experiences. As a research contribution, we suggested a generic architecture (MDVI). The MDVI architecture can be used for implementing mobile device applications, that interact with integrated systems in vehicles. In order to evaluate the architecture design, we developed a prototype application (KITT) by following an adapted iterative approach that allowed for frequent improvements of the architecture. While the prototype application was not implemented in its entirety, it could still show the potential of the MDVI architecture. As we went through the three iteration stages we continuously made addition to the MDVI architecture and improvements to the functionality already in place. The process followed is depicted in Figure 7.1



Contribution Domain
Contribution Type
Defining the Stakeholders
Identifying the Quality Attributes
Determining the Architectural Pattern
Communicating with the vehicle
Ensuring Security
Including Vehicle information
Building for Entertainment
Communicating with the world
Storing the data
Research
Architecture
MDVI
Practice
Prototype

Figure 7.1 *Iterative research process outcome*

The purpose of the first iteration was to design a first draft for the MDVI architecture, based on the literature study conducted. We sought to learn which architecture pattern is appropriate for our purpose and which guidelines it should follow. By developing a prototype application, we expected to receive outputs that could be used to improve the architecture.

The literature study carried out gave us a better understanding about the field of telematics and what it involved in terms of technologies used and features that are commonly utilized. We also learned to what level of significance the Android architecture has for the architecture we wanted to propose. The wireless technologies were investigated and taken into consideration if they could be conveyed in our proposed architecture.

We realized that to not limit the architecture, by making it available only to regular users, we needed an extension allowing less restricted use. This could enable e.g. service technicians to access most of the information in the vehicle system and get a detailed status report. Making this restriction distinction gave us a direction of the further development in terms of functionality and usability. Usability is addressed in the MDVI architecture as the User and Service Technician modules in the User Interface tier.

By creating use case scenarios we were able to make a selection of quality attributes that aided us in the design and development of the architecture and the prototype. They also add a point of reference to keep the focus on our objectives. To structure our architecture we found the three-tier pattern appropriate, as it corresponded well with modules separation, security handling and persistent storage.

From the prototype implementation itself we learned that in order to make the integration of multiple modules easier we needed some sort of connection point, the core module. Development further revealed that we could use Android's communication manager to set up connections to a vehicle. Since different vehicles use different communication protocols, we choose to abstract away from vehicle side implementation and focus on the mobile device side, using a simulator to simulate the vehicle.

In the first iteration we acknowledge security as one of our quality attributes, which only became more evident when we started to access actual data from the vehicle's systems during the second iteration. We also realized that aside from authentication for using the application, Application Security module should handle authentication with the vehicle gateway and encryption of transferred data. By adding the Application Security module, we addressed the security quality attribute that was assigned to the MDVI architecture.

When discussing the Vehicle Monitoring module, we understood that it would be necessary to add two additional modules, Vehicle Service Diagnostics and Log Manager, to facilitate the separation of the regular user and service technician privileges. The Log Manager module gave us a way to more conveniently handle messages from Message Handling module, as notifications intended for Vehicle Monitoring and Vehicle Service Diagnostics have to be formatted differently.

During the implementation of the Vehicle Monitoring module we further learned, that in order to enable the ability to constantly request, and eventually also send data to the vehicle while simultaneously have the application process other tasks, we needed to put ongoing or computationally expensive processes in their own threads. Not doing so caused the application to freeze or behave unexpectedly.

From the third iteration we understood that we needed to integrate some external service functionality to the MDVI architecture. When we began the development phase of the third iteration we learned that there were specific APIs for both Google Maps and Facebook to facilitate implementation of such functionality. When we eventually implemented the Database Coordinator, it enabled the vehicle data to be used in more useful ways than to just simply display them as part of the Vehicle Monitoring module. By saving values over time we could calculate value average, show statistics, and

through that also create a better chance for a more accurate vehicle diagnose, in case of a failure.
The MDVI architecture can be used by developers to implement similar mobile device applications for vehicle interaction. Since the design of the MDVI architecture is aimed for reusability, it could be extended for interaction with embedded systems other than vehicles. The inclusion of diagnostics aspect to the architecture, enable developers to focus on development of service and diagnostic applications for current generation mobile devices. Specifically for our industry partners, the knowledge, experience and ideas gained from the implementation of the KITT prototype will be partly used as a base for their application development. It may also further be used as input to research regarding how mobile devices can work in conjunction with their vehicles onboard systems.

Referring back to Vaishnavi & Kuechler (2005), our study may be categorized as between "firm" and "loose ends". It is evident to us that further research in the area of mobile device platforms and vehicle interaction is still needed. Such research could help improve and extend the MDVI architecture. We suggest that future steps for evaluating the MDVI architecture focus on other mobile device platforms such as Apple's iOS and Microsoft's Windows Phone 7. This will strength the architecture completeness by validating it for other mobile platforms, and may expose features or flaws not identified through implementation in Android.

As we discuss in this report as part of the architecture evaluation, further research could also look to the integration of peer-to-peer solutions, as well as support remote interaction through a server. By taking a scenario from the vehicle-to-vehicle domain, where messages about road safety issues and accidents can be communicated directly between vehicles. We can envision how mobile devices may be used in the same manner, utilizing peer-to-peer technologies for communicating in critical situations or maybe just for comparing vehicle values between friends. Remote interaction through a server creates more opportunities for the user. Imagine a user being able to connect to his electric vehicle from his office, check vehicle battery status or upload a route planned in his tablet PC.

Further research as we suggest will not only contribute to the problem domain we address in this study, but can also add knowledge to related domains that can exploit such mobile device-to-embedded system interaction.

Acknowledgements

We would like to thank Carl Magnus Olsson, Gerardo Schneider and Henrik Fagrell for the supervision, ideas and help. We would also like to thank Ulrik Eklund, Amadeus Hein & Fredrik Bengtsson for the extra support and guidance.

References

API (computer programming) -- Britannica Online Encyclopedia. 2011. API (computer programming) -- Britannica Online Encyclopedia. [ONLINE] Available at: http://www.britannica.com/EBchecked/topic/1472947/API. [Accessed 16 April 2011].

Bass, L., Clements, P., Kazman., R., 2003. Software Architecture in Practice. 2nd ed. Reading, MA: Addison-Wesley Professional.

Bilchev, G., Marston, D., Hristov, N., Peytchev, E., Wall, N., 2004. Traffimatics - intelligent co-operative vehicle highway systems. Bt Technology Journal, 22 (3), pp. 73-83.

Day, J.D., Zimmermann, H., 1983. The OSI Reference Model. Proceedings of the IEEE, Vol. 71, No. 12., pp. 1334-1340.

Dhawan, S., “Analogy of Promising Wireless Technologies on Different Frequencies: Bluetooth, WiFi, and WiMAX,” Wireless Broadband and Ultra Wideband Communications, 2007. AusWireless 2007. The 2nd International Conference on , vol., no., pp.14, 27-30 Aug. 2007

Gehlen, G., Weiss, E., Lukas, S., Rokitansky, C.H., Walke, B., Architecture of a Vehicle Communication Gateway for Media Independent Handover, In Proceedings of the 3rd International Workshop on Intelligent Transportation (WIT2006), pp. 205-209, Hamburg, Germany, March 2006

Grymek, L., Singh, S., Pattipati, K., , “Vehicular dependence adds to telematics’ allure,” Potentials, IEEE, vol.26, no.2, pp.12-16, March-April 2007

Hsu, R.C., Chen, L.R., “An Integrated Embedded System Architecture for In-Vehicle Telematics and Infotainment System,” Industrial Electronics, 2005. ISIE 2005. Proceedings of the IEEE International Symposium on , vol.4, no., pp. 1409- 1414, June 20-23, 2005

Hubaux, J.P., Capkun, S., Luo, J., “The security and privacy of smart vehicles,” Security & Privacy, IEEE, vol.2, no.3, pp.49-55, May-June 2004

Hossain, E., Chow, G., Leung C.M.Victor, McLeod, R.D., Misic, J., Wong, W.S. Vincent, Yang, O., Vehicular telematics over heterogeneous wireless networks: A survey, Computer Communications, Volume 33, Issue 7, 3 May 2010, Pages 775-793,

Juliussen, E., 2003. The future of Automotive Telematics. Business Briefing: Global Automotive Manufacturing & Technology.

Karimi, A., Olsson, J., Rydell, J., 2004. A Software Architecture Approach to Remote Vehicle Diagnostics. M.Sc. Gothenburg University.

Kuschel, J., 2008. The Vehicle Ecosystem. In: León, G., Bernardos, A., Casar, J., Kautz, K., De Gross, J. Open IT-Based Innovation: Moving Towards Cooperative IT Transfer and Knowledge Diffusion. Boston: Springer. p309-322.

Mahmoud, Q.H., “Integrating mobile devices into the computer science curriculum,” Frontiers in Education Conference, 2008. FIE 2008. 38th Annual , vol., no., pp.S3E-17-S3E-22, 22-25 Oct. 2008

Nolte, T., Hansson, H., Bello, L.L., “Automotive communications-past, current and future” Emerging Technologies and Factory Automation, 2005. ETFA 2005. 10th IEEE Conference on , vol.1, no., pp.8 pp.-992, 19-22 Sept. 2005

Pei, L., Chen, R., Chen, Y., Leppäkoski, H., Perttula, A., 2009. Indoor/Outdoor Seamless Positioning Technologies Integrated on Smart Phone. IEEE, 12, 141-145.

Poul H., Andersen, J.A., Mathews, M.R., Integrating private transport into renewable energy policy: The strategy of creating intelligent recharging grids for electric vehicles, Energy Policy, Volume 37, Issue 7, July 2009, pp.2481-2486.

Putrus, G.A., Suwanapingkarl, P., Johnston, D., Bentley, E.C., Narayana, M., “Impact of electric vehicles on power distribution networks, “Vehicle Power and Propulsion Conference, 2009. VPPC ‘09. IEEE, vol., no., pp.827-831, 7-10 Sept. 2009

Radiocommunication Sector (ITU-R) - ITU global standard for international mobile telecommunications ‘IMT-Advanced’. 2011. Radiocommunication Sector (ITU-R)

- ITU global standard for international mobile telecommunications ‘IMT-Advanced’. [ONLINE] Available at: http://www.itu.int/ITU-R/index.asp?category=information&rlink=imt-advanced=en. [Accessed 31 March 2011].

Runeson, P., Höst, M. 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering, 14(2), pp.131-164.

Seada, K., “Mobile Devices as an Extension to the Wireless Infrastructure,” Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE , vol., no., pp.1-5, 9-12 Jan. 2010

Smartphone statistics for Q2 2008, http://www.canalys.com/pr/2008/r2008082.pdf, 15 August 2008, verified 15 January 2009.

Socket | Android Developers. 2011. Socket | Android Developers. [ONLINE] Available at: http://developer.android.com/reference/java/net/Socket.html. [Accessed 16 April 2011].

Vassilaras, S., Yovanof, G., Wireless Innovations as Enablers for Complex &amp; Dynamic Artificial Systems, Journal Name: Wireless Personal Communications, Cover Date: 2010-05-01, Publisher: Springer Netherlands, Volume: 53, Issue: 3, pp.365-393.

Vaishnavi, V., Kuechler, W., 2005. “Design Research in Information Systems” January 20, 2004, last updated August 16, 2009. URL: http://desrist.org/design-research-in-information-systems

Vaughan-Nichols, S.J., “OSs battle in the smart-phone market,” Computer , vol.36, no.6, pp. 10- 12, June 2003

What is Android? | Android Developers. 2011. What is Android? | Android Developers. [ONLINE] Available at: http://developer.android.com/guide/basics/what-is-android.html. [Accessed 11 May 2011].
Willie, T., “Broadband Over Power Lines,” Power Line Communications and Its Applications, 2006 IEEE International Symposium on , vol., no., pp. 1, 26-29 March 2006
Y.Wu, P.Chen, Z.Hu, C.Chang, G.Lee, W.Yu, 2009. A mobile health monitoring system using RFID ring-type pulse sensor. IEEE, 136, 317-322.

Appendix A

Use case scenarios

Use case #2 **Check the fuel level**
Brief description: User gets information about vehicle status and fuel level in his smartphone display.
Flow of events:
<u>Basic Flow:</u>
- User clicks on "Vehicle Status" button in the application menu
- User receives data regarding vehicle status and fuel level

<u>Alternative flow:</u>
- User clicks on "Vehicle Status" button in the application menu
- System response time is more than 5s
- User receives an error message that the fuel level cannot be read since it has not been received from the vehicle because an internal error on the vehicle side.

Precondition: User's smartphone is connected to the vehicle via WiFi. User starts the vehicle application in his smartphone and is logged in and his identity is authenticated as the vehicle's user.
Post-condition: User gets data from the vehicle to his smartphone regarding fuel level.

Use case #3 **Download travelled route from the vehicle**
Brief description: User gets the travelled route that has been done in the vehicle, downloaded to his smartphone.
Flow of events:
<u>Basic Flow:</u>
- User clicks on "Download Route" in the application menu.
- Route information is downloaded to the smartphone
- New route information is available in the list of routes

<u>Alternative flow:</u>
- User clicks on "Download Route" in the application menu
- Route information starts downloading to the smartphone but the connection gets interrupted
- User fails to receive the total route data for the travelled journey, an error message is displayed

Precondition: User's smartphone is connected to the vehicle via WiFi. User starts the vehicle application in his smartphone and is logged in and his identity is authenticated as the vehicle's user.
Post-condition: User receives data regarding the travelled route from the vehicle to his smartphone.

Use case #4 **Notifications received from the system**
Brief description: User logs in to the system and gets a system notification about the vehicle's fuel level.
Flow of events:
<u>Basic Flow:</u>
- User starts the vehicle application in his smartphone
- User logs in as the owner of the vehicle and his identity is authenticated
- User receives a notification in the display regarding fuel level

<u>Alternative flow:</u>
- User starts the vehicle application in his smartphone
- User logs in as the owner of the vehicle and his identity is authenticated
- An error message is displayed because the connection to the vehicle has been lost
- User tries to reconnect to the vehicle

Precondition: User's smartphone is connected to the vehicle via WiFi.
Post-condition: User receives a notification from the system.

Use case #5 **Change pin code for log-in**
Brief description: User wants to have a different log in-code and therefore changes it.
Flow of events:
<u>Basic Flow:</u>
- The user clicks on "Settings" in the application menu
- The different settings show up in the display
- The user clicks on "Change Pin code" in the settings menu
- The user enters the old pin code together with the new desired pin code, and presses "OK"

<u>Alternative flow:</u>
- The user clicks on "Settings" in the application menu
- The different settings show up in the display
- The user clicks on "Change Pin code" in the settings menu
- The user enters the old pin code together with the new desired pin code, and presses "OK"
- An error message is display saying that the old pin code entered is incorrect, the service technician has to reenter the old pin code and the new desired pin code

Precondition: User's smartphone is connected to the vehicle via WiFi. Service technician starts the vehicle application in his smartphone and is logged in and authenticated as a service user.
Post-condition: The pin code to log in to the vehicle application has been changed and the old pin code has been removed.

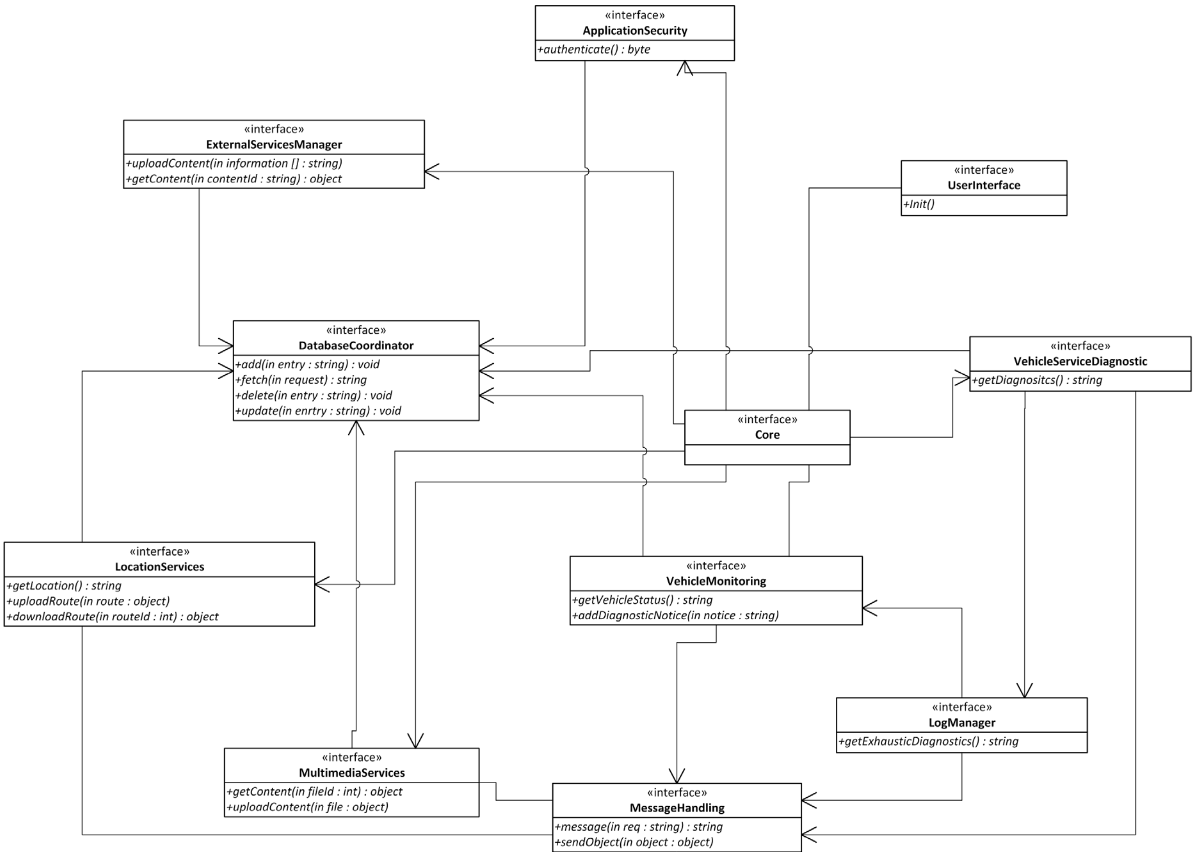
Use case #6 **Upload information to remote service**
Brief description: User wants to upload data to a remote service.
Flow of events:
<u>Basic Flow:</u>
- User clicks on "Upload Data" in the application menu
- Two options are available, "Upload to service center" or "Upload to remote service"
- User clicks on "Upload to remote service"
- Different remote services are displayed as options on next display
- The user selects Facebook as a remote service and clicks "Upload"
- A message displays on the screen, that the upload has been successful

Alternative flow:
- User clicks on "Upload Data" in the application menu
- Two options are available, "Upload to service center" or "Upload to remote service"
- User clicks on "Upload to remote service"
- Different remote services are displayed as options on next display
- The user selects Facebook as a remote service and clicks "Upload"
- Connection to Facebook cannot be established, the service is down

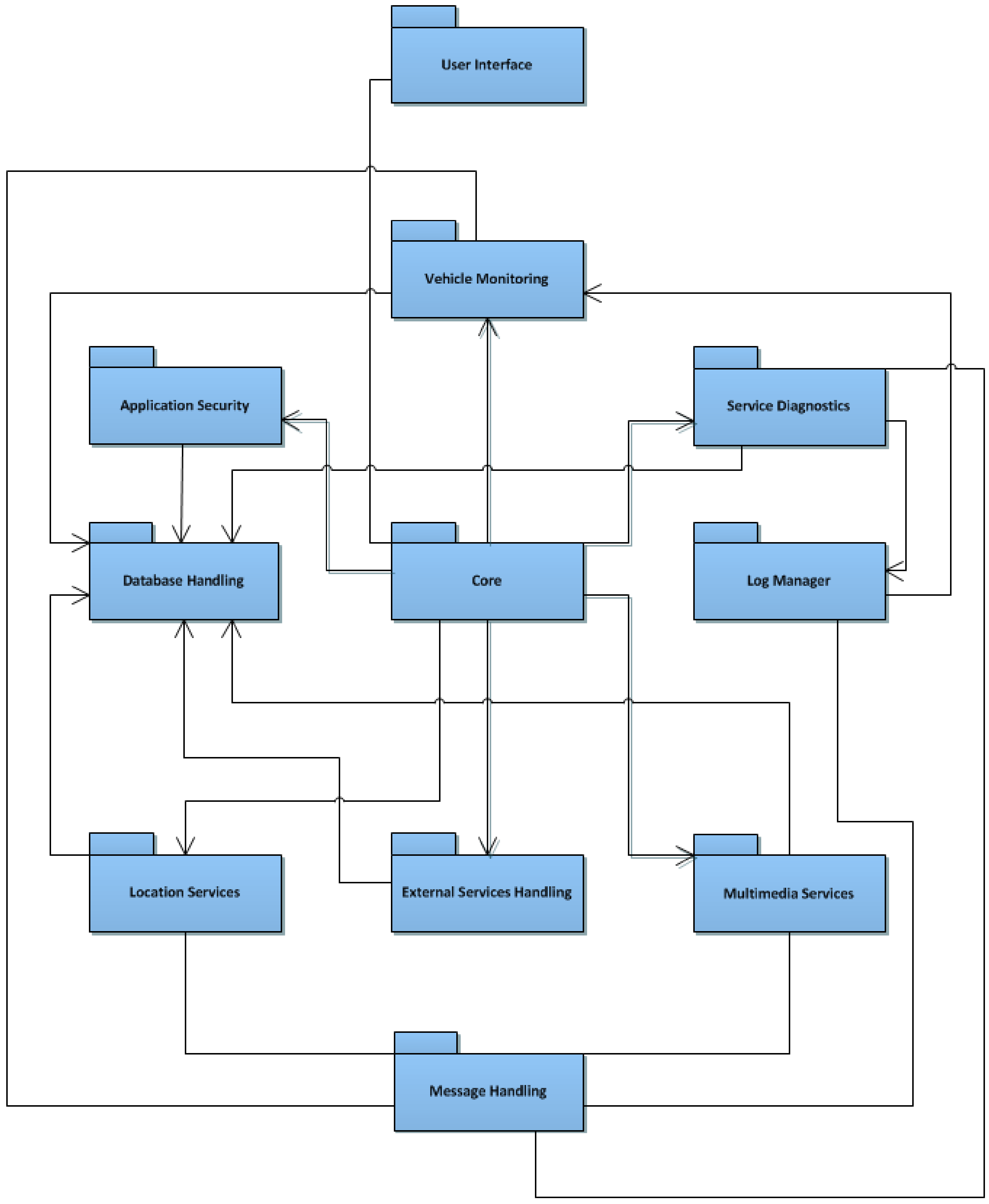
Precondition: User starts the vehicle application and is logged in and authenticated as a service technician.
Post-condition: The information is uploaded to some remote service (e.g. like Facebook).

Appendix B

MDVI Modules Interface Diagram

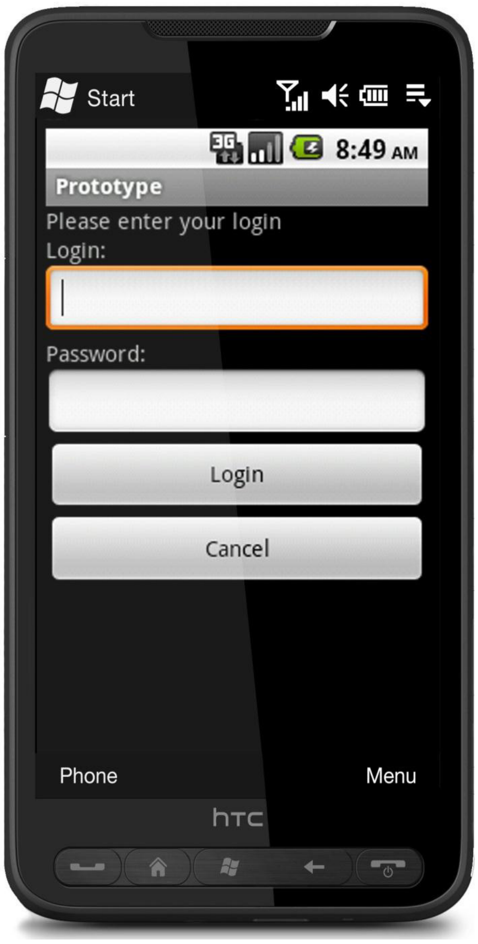

«interface»
ApplicationSecurity
+authenticate() : byte
«interface»
ExternalServicesManager
+uploadContent(in information [] : string)
+getContent(in contentId : string) : object
«interface»
UserInterface
+Init()
«interface»
DatabaseCoordinator
+add(in entry : string) : void
+fetch(in request) : string
+delete(in entry : string) : void
+update(in enrtry : string) : void
«interface»
VehicleServiceDiagnostic
+getDiagnositcs() : string
«interface»
Core
«interface»
LocationServices
+getLocation() : string
+uploadRoute(in route : object)
+downloadRoute(in routeId : int) : object
«interface»
VehicleMonitoring
+getVehicleStatus() : string
+addDiagnosticNotice(in notice : string)
«interface»
LogManager
+getExhausticDiagnostics() : string
«interface»
MultimediaServices
+getContent(in fileId : int) : object
+uploadContent(in file : object)
«interface»
MessageHandling
+message(in req : string) : string
+sendObject(in object : object)

MDVI Package Diagram



Appendix C

KITT Application Prototype



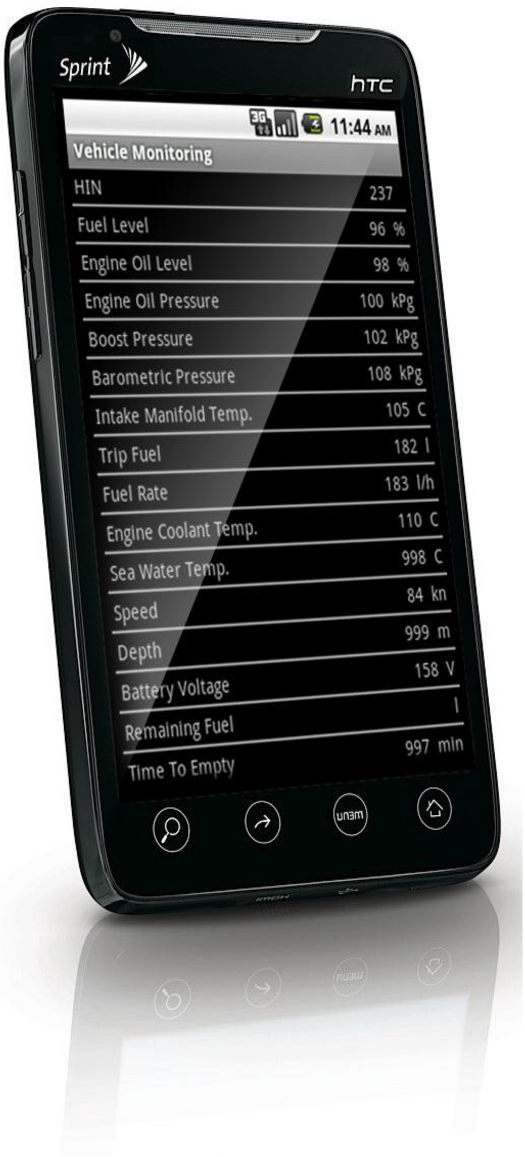
Please enter your login

Login:

Password:

Login

Cancel

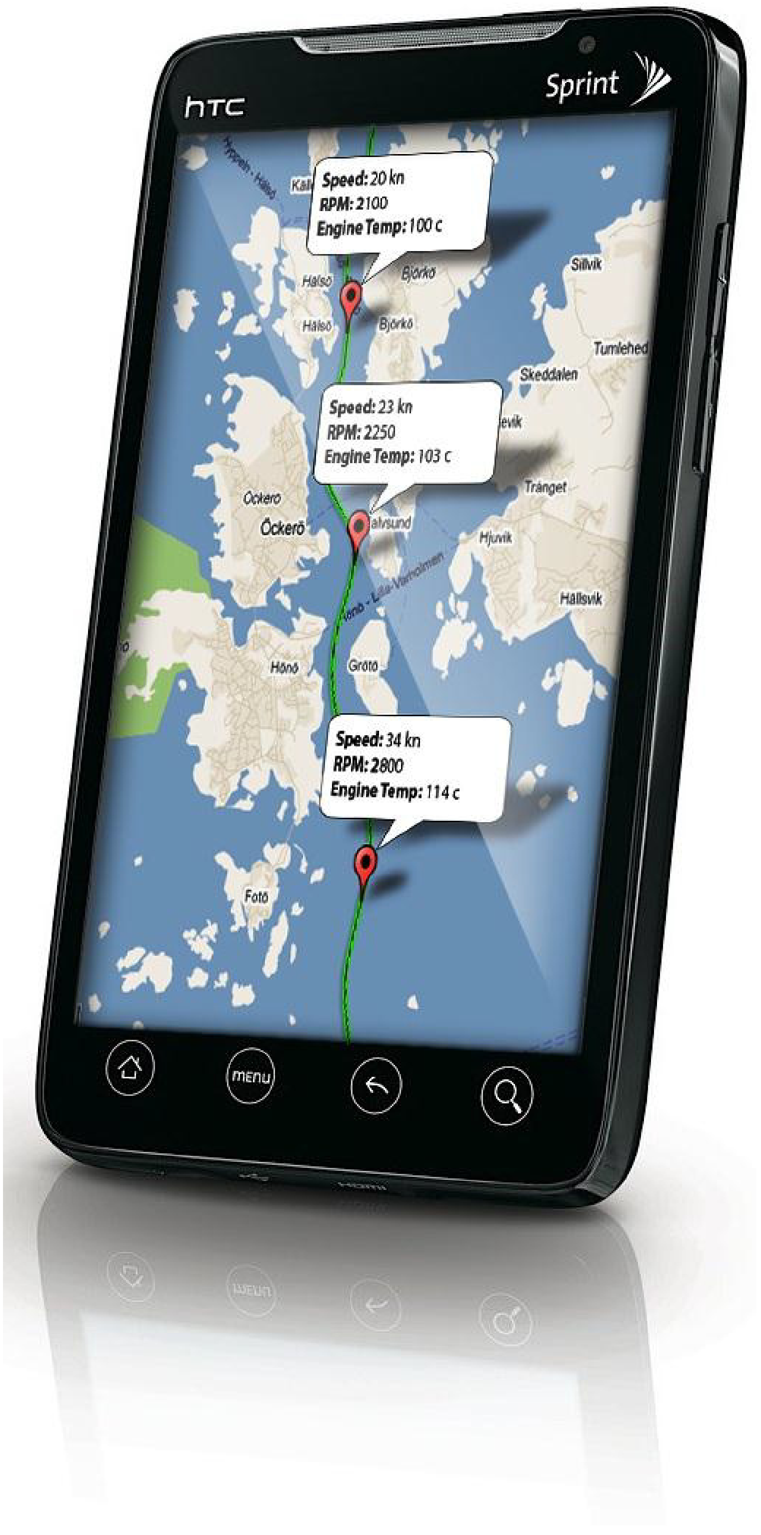


Vehicle Monitoring

HIN	237
Fuel Level	96 %
Engine Oil Level	98 %
Engine Oil Pressure	100 kPg
Boost Pressure	102 kPg
Barometric Pressure	108 kPg
Intake Manifold Temp.	105 C
Trip Fuel	182 l
Fuel Rate	183 l/h
Engine Coolant Temp.	110 C
Sea Water Temp.	998 C
Speed	84 kn
Depth	999 m
Battery Voltage	158 V
Remaining Fuel	l
Time To Empty	997 min

Appendix D

Google Maps Concept


htc
Sprint
Speed: 20 kn
RPM: 2100
Engine Temp: 100 c
Speed: 23 kn
RPM: 2250
Engine Temp: 103 c
Speed: 34 kn
RPM: 2800
Engine Temp: 114 c
Sillvik
Tumlehed
Skeddalen
Tranget
Hjuvik
Hällsvik
Öckerö
Hönö
Grötö
Fotö
Björkö
Hälsö
menu